IN THE SPECIFICATION:

Each paragraph is listed in its amended form per 37 CFR 1.121(b)(1)(ii).

1. Please replace the third paragraph on page 5 with the following paragraph:

   State information is usually saved in memory and may be fetched sequentially every time the functional unit is run. In a multiprocessor system with separate memory banks and caches, the processor running the functional unit may have idle time while other parts of the application are running on different processors. During this dead time, the processor could be fetching the state information from memory into ~~it's~~ its cache assuming it knows which state information it needs. It could know this by using call prediction and would save all the time necessary to fetch the state information from memory. This could result in a significant speedup. Similar speed up could be achieved for other hardware units capable of parallel execution.

2. Please replace the second paragraph on page 10 with the following paragraph:

   There are a few characteristics of applications that can take advantage of a caller prediction model for prefetching node instance data. In one embodiment, applications should not execute in hard real-time since the prediction logic may be wrong, which may result in jitter. In one embodiment, applications for run-time caller prediction may include applications where quality of service (QoS) can vary depending on how fast the application is running. Buttazzo et al. point out that these applications are common due to the non-deterministic behavior of common low-level processor architecture components, such as caching, prefetching, and direct memory access (DMA) transfers. Buttazzo's work suggests voice sampling, image acquisition, sound generation, data compression, video playback, and certain feedback control systems as application domains that can function at varying QoS depending on the execution rate of the application.

3. Please replace the fourth paragraph on page 14 with the following paragraph:

   In one embodiment, each cell in the call history table 11 may be managed by update logic 56 that may operate to increment, i.e., use table update 62, when a prediction 20A is correct and decrement when a prediction 20A is incorrect. For example, in order to generate a prediction 20A, the column values across the call history table 11 may be compared (12A) to find the greatest value.

4. Please replace the last paragraph on page 15 with the following paragraph:

> In one embodiment, after the table lookup, the call prediction may become the counter with the largest value. For example, ~~for~~ if a history of past callers is 231, then the index may become 15 and the call prediction may predict the caller 2 as the next caller. In one embodiment, the table may be updated if the call prediction was correct.

5. Please replace the fifth paragraph on page 16 with the following paragraph:

> For example, for a past call history of *12341234*, the first equality is true, and the periodic caller prediction may select *4* as the prediction 20. Similarly, for a history of *02345602*, the third equality is true, and periodic caller prediction may select *6* as the prediction 20. These comparisons assume that the period of the calling sequence is contained in the call history register 10, and that call prediction latency is equal to the length of the call history register 10 if the period or the calling sequence changes.